文章编号：2095-2163(2023)03-0064-05 中图分类号：TP311.5 文献标志码：A

基于微服务架构的大宗商品 ERP 仓储系统的研究

姚 砺，徐梦娜，张海路，付 帅
（东华大学 计算机科学与技术学院，上海 201620）

摘 要：当前针对大宗商品企业开发的 ERP 仓储信息管理系统大多为单体架构，存在功能老旧[1]、安全性低且业务模块耦合度高的问题[2]。本文涉及的跨国公司主营大宗商品，业务复杂且涵盖海内外多地、更新频繁、交易量巨大，单体架构下的系统无法满足大型公司复杂的业务需求、严格的安全需求、高并发场景下的可用性以及数据一致性。为解决以上问题，本系统以微服务架构为基础，基于 Spring Cloud 框架，结合服务熔断限流、服务无状态化设计、分布式事务管理等技术，设计开发了低耦合、易扩展、高并发，高可用、高安全性的分布式系统[3]。本系统现已交付某大型跨国企业使用，并取得良好效果。

关键词：微服务架构；大宗商品仓储管理系统；Spring Cloud；分布式事务

Research on bulk commodity ERP storage system based on microservice architecture

YAO Li, XU Mengna, ZHANG Hailu, FU Shuai
(College of Computer Science and Technology, Donghua University, Shanghai 201620, China)

【Abstract】 Presently, most ERP warehouse information management systems developed for bulk commodity enterprises are single architecture, which has the problems of old function[1], low security and high coupling of business modules[2]. The multinational companies involved in this paper are mainly engaged in bulk commodities. Their businesses are complex and cover many places at home and abroad, with frequent updates and huge transaction volume. The system under the single architecture can not meet the complex business requirements, strict security requirements, availability and data consistency of large companies in high concurrency scenarios. In order to solve the above problems, based on the microservice architecture and the Spring Cloud framework, combined with the technologies of service fusing and current limiting, service stateless design and distributed transaction management, this system designs and develops a distributed system with low coupling, easy expansion, high concurrency, high availability and high security[3]. The system has been delivered to a large multinational enterprise and achieved good results.

【Key words】 microservice architecture; bulk commodity ERP storage system; Spring Cloud; distributed transaction

0 引 言

以 ERP 系统为主的信息管理系统在企业中得到广泛运用，但是当前市场上并没有针对大宗商品交易的 ERP 系统，其交易的特殊性在于大宗商品交易的数量和金额巨大，且业务流程复杂多样，通用的 ERP 系统并不能满足该类企业的业务需求。另外，传统的信息管理系统通常采用一体式的单体架构[4]进行开发，结构冗余、功能老旧单一且不利于更新换代，只能够应对企业在初级阶段所面临的业务问题。随着互联网技术和企业发展的多维协作和和深度融合，仓储管理系统面临着诸如业务逻辑越发复杂、业务需求更新换代更快、企业信息安全性无法保证等诸多问题[5]。

目前，ERP 系统开发多基于单体架构，使用 SSM 框架（Spring、Spring MVC、Mybatis）将整个应用作为一体进行开发[6]。其中，通过 Spring 框架进行开发时，首先需要进行复杂的 xml 配置，配置易出错且易影响开发效率。当配置过于复杂时，还会导致容器的响应速度下降。Spring MVC 作为 MVC 架构的代表技术而言，在业务规模小时有良好的表现，但在随着业务不断复杂、访问量不断增加时，该架构显露出并发请求响应时间过长、新增部署服务器操作繁琐等严重问题[7]。本文所涉及的跨国公司业务

作者简介：姚 砺（1967-），男，博士，副教授，硕士生导师，主要研究方向：软件测试技术、图像处理；徐梦娜（1997-），女，硕士研究生，主要研究方向：计算机应用、软件开发；张海路（1998-），男，硕士研究生，主要研究方向：计算机应用、软件开发；付 帅（1997-），男，硕士研究生，主要研究方向：计算机应用、软件开发。

收稿日期：2022-04-22

复杂、交易时并发量大，显然当前常用的 ERP 开发模式无法搭建一个符合本公司需求的高可用、高安全性以及高伸缩性的仓库管理系统。

针对目前基于单体架构的 ERP 系统设计存在的问题，本文实现了基于微服务架构的面向大宗商品的仓储管理系统。系统基于微服务划分原则，将整体业务拆分为独立的微服务降低系统耦合性。采用 Spring Cloud 框架，结合服务熔断限流、服务无状态化设计、事务管理服务等技术搭建了一个高并发，高可用、高安全性、高伸缩性以及事务一致性的分布式系统，并在实际应用中取得了优异表现，也为其他具有同类业务需求的大型公司提供了微服务架构下仓储管理系统的开发思路[8]。

1　系统需求分析

1.1　业务需求

对于主营原木大宗商品的跨国公司而言，在进行仓储管理时存在商品品种繁多、不同品类间存储或处理流程多样、存储的数量庞大、涉及跨地区流转等特点。针对以上特点，设计时将业务拆分为入库、检测熏蒸、装箱转运等模块，如图 1 所示。另外，在大型跨国公司中，公司业务的严谨性也尤为重要，每一项流程都需要高级权限进行审核，审核之后存在一定情况需要人为回退的为反审。由于原木商品进行跨国交易时，涉及到的海关政策、实时汇率等突发性情况较多，因此反审功能极为重要，需要针对每一个流程进行定制化的回退，是本系统实现的难点。系统关键业务流程如图 1 所示。

1.2　可用性需求

由于原木商品交易数据极大、并发高、交易涉及面广，系统的高并发、高可用是系统实现的难点之一；其次，跨国公司的合作商与员工众多，因此系统需要更严格的权限认证与访问控制，如何达到系统的高安全性也是系统实现的难点；最后，在仓储系统与其他业务子系统进行跨服务、跨系统调用时，保证系统的事务一致性同样是系统实现的难点。

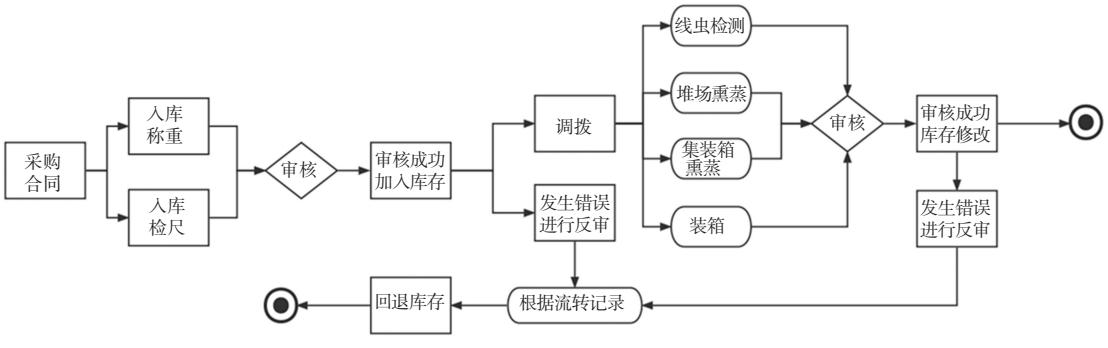

图 1　关键业务流程图

Fig. 1　Key business implementation process

2　基于微服务架构的仓储管理系统架构设计

当前很多的 ERP 系统都采用单体架构。所谓单体架构，即指将整个软件系统的功能模块及运行数据等作为整体看待，再进行统一地设计、开发、打包及部署运行[9]。在单体架构式应用中，所有的业务逻辑作为整体进行设计，一旦企业的业务需求发生更新，则需要从底层进行修改，开发成本耗费巨大。

本文涉及的跨国公司业务功能复杂，若以单体架构进行开发会导致整个系统逻辑过于复杂，维护与更新成本巨大。另外，该公司交易量大、并发高，单体架构应用在高并发场景下会出现服务崩溃、响应时间过长等致命问题。所以，本文的仓储 ERP 系统并不适合单体架构进行开发。而属于分布式架构的微服务能够将整体应用进行拆分，模块之间独立更新，能够更好地适应企业当下快速变化的业务需求。另外，对于一个大型跨国公司而言，内部信息和数据的安全性至关重要，仅仅依靠微服务架构无法保证系统的信息数据安全性。

因此，本系统基于微服务架构实现系统的高伸缩性、结合熔断限流和服务无状态化设计实现高可用性、结合分布式事务管理以保证数据一致性来实现仓储系统。

2.1　微服务架构简介

微服务架构（Micro Services Architecture，MSA）是指根据应用系统的业务需求，通过对预定义的微服务进行重组而形成企业级应用的分布式体系结构[10]。微服务架构的基本思想是将传统的单体应用按业务功能拆分为一系列可被独立设计、开发、部署、运维的软件服务单元，服务间彼此配合、相互协

作以实现最终价值[11]。相比较于传统的单体架构，微服务架构模式具有以下众多优势：

(1)开发效率高。每个微服务技术多元化，有效降低维护风险[12]，开发时每个部分可以选取不同的技术，只要提供统一的对外接口即可，极大提高了开发效率。

(2)低耦合性与强扩展性。微服务架构属于分布式架构，每个微服务独立开发部署互不影响，服务间通过 RPC 进行调用[13]，耦合度低。单个服务升级更新不影响总体，具有强大的横向扩展性。

(3)高可用性。根据实际需要，对部分微服务进行集群部署，解决了随用户数量上升带来的高并发问题。

2.2 系统架构设计

针对大宗商品交易流程复杂、需求多变、交易时数量庞大等特殊性，本系统基于微服务架构，搭建一个低耦合、易扩展、高并发、高可用、高安全性的分布式系统。目前，使用较主流的 4 种微服务框架、即 Dubbo、Motan、gRPC 和 Spring Cloud，其中 Dubbo、Motan 属于服务治理型 RPC 框架提供了全面的服务治理功能，gRPC 虽然没有全面的服务治理功能但是提供了多种语言的接口支持[14]。与其他 3 种比较而言，Spring Cloud 不仅提供了服务治理功能，同时还包含服务网关 Gateway、服务熔断 Sentinel、服务调用 Feign 等搭建微服务所需要的组件，通过负载均衡、服务无状态设计、网关鉴权等技术能够满足本系统所需的并发性、可用性与安全性。基于 Spring Cloud 以上优势，本系统选取 Spring Cloud 2020 快速搭建微服务[15]。系统架构如图 2 所示。由图 2 分析可知，对该系统的功能特性可做阐释分述如下。

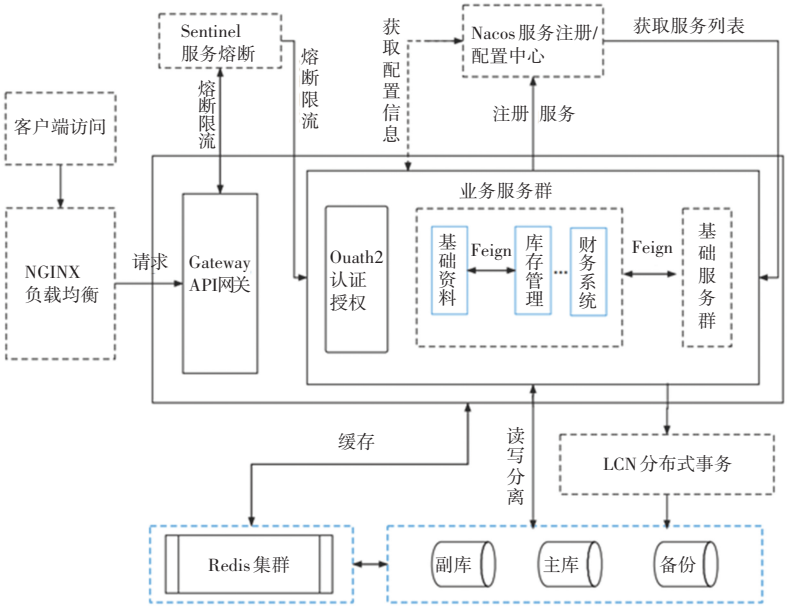

图 2 系统架构图

Fig. 2 The architecture of the system

(1)伸缩性：基于公司业务复杂且更新快的特点，将公司的业务服务群分为基础资料、库存管理、货品运输等 5 个，每个业务服务功能与设计独立，便于扩展与升级，具有良好的伸缩性。

(2)安全性：基于跨国公司常涉及海内交易且用户群广泛，为避免恶意的网络攻击、非法入侵，系统引入 Oauth2.0 作为安全解决方案提高系统安全性[16]。

(3)高效性：为解决微服务下的服务管理问题，将所有服务在 Nacos 注册与配置中心进行注册后，对这些服务进行独立配置、部署及管理。利用动态配置，以集中和动态的方式管理各个服务的配置信息，无需重新部署服务，使配置管理具有高效性。

(4)高并发可用性：基于公司交易商品的交易量大，瞬间的大量请求会导致服务瘫痪，造成服务雪崩。为解决以上问题，整合 Sentinel[17] 做到流量控制、熔断降级等来提升服务的稳定性、保持服务在高并发性能。

(5)事务完整性：基于公司业务繁多，在系统内划分多个业务模块，必然会涉及到跨服务调用导致数据的不一致性。为解决此类事务问题、保证事务的一致性，本系统引入 LCN 分布式事务框架实现事务完整性[18]。

3　系统关键技术

针对大宗商品市场用户覆盖面广、跨国交易复杂、交易时用户与货品数量庞大等特点，如何搭建一个高可用、高安全性以及具有事务一致性的分布式系统是本系统的关键。

3.1　安全性与高可用性

（1）无状态登录实现登录安全性。在大型企业中，公司的机密信息显然需要更严格的控制。本文涉及的跨国公司业务广泛、覆盖几十个国家和地区、用户的使用环境与信息传输都存在风险。因此提供安全可靠的登录认证服务也是本系统实现的关键点之一。传统的登录认证在当前客户端登录后，如需访问另外的服务提供者，则需要当前客户端的用户密码来获取当前的用户信息，此时存在密码暴露的风险。因此为解决以上风险，本系统基于 Spring Oauth2.0 设计思想，在客户端与服务提供者间加入授权层，客户端无法直接登录服务提供者[19]，而是通过授权层获取设有有效期的令牌（Token），客户端携带令牌访问，授权层拦截 token 进行检查处理和查询用户信息[20]，把无状态的 token 转化成用户信息，实现用户无状态登录，具有良好的安全性。无状态登录的认证过程如图 3 所示。

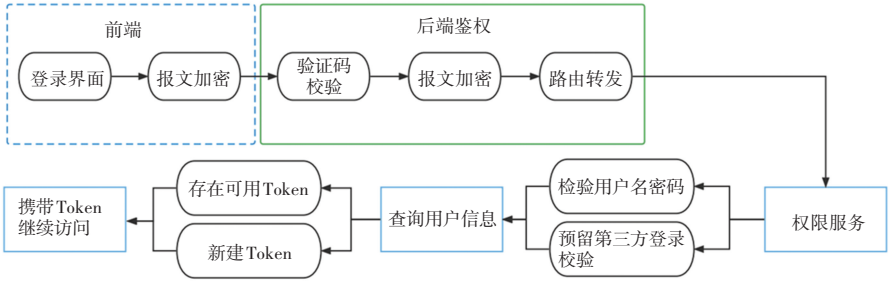

图 3　无状态登录的认证过程图

Fig. 3　Authentication process of stateless entry

（2）服务层无状态实现服务层高可用性。本文涉及的跨国公司用户量庞大、业务量高，当同时出现大量访问时，单个微服务性能可能会遭遇瓶颈[21]。因此，实现服务层的高可用性也是本系统的关键之一。针对以上问题，本系统内使用无状态登录可以使得服务器端无需保留大量用户数据，减轻服务端压力，实现服务层无状态。服务层无状态化设计使集群中的节点可彼此替代，任何节点宕机都不会导致系统停止服务，实现了服务层的高可用性。

3.2　事务完整性

在微服务架构下，各功能都按照微服务划分原则[16]分为独立的微服务，但在进行跨服务调用时，会涉及到一次操作需要多个系统协同进行，不管是对同一个数据库的操作、还是多个数据库，此时则要保证这些操作都要同时成功或同时失败[22]。这样一来，Spring 自带的事务会失效，造成系统数据的不一致性。

对于本文的跨国公司而言，业务逻辑复杂需要将业务拆分为多个微服务，存在跨服务和跨数据库的事务操作。因此，解决微服务架构下的事务完整性也是本系统的关键。为此，本文将 LCN 框架引入微服务架构中，以保证本系统在跨服务场景下的数据一致性。

LCN 框架的基本思想是不创建事务、不操作事务，通过协调事务来达到事务一致性[21]。在进行跨服务操作时，TxManager 作为事务协调者独立在业务服务之外，不嵌入业务代码中，因此与业务服务的耦合性低。并且 TxManager 能够兼容 Nacos 注册中心并进行服务注册，便于独立部署与管理。基于 LCN 框架对业务的入侵性低、对第三方框架的兼容性强等优势，选取 LCN 框架能够更好地实现本系统的数据一致性。

对于交易量大、并发高的跨国公司而言，事务管理的高可用性同样至关重要。TxManager 集群可以与 Redis 相结合，将 TxManager 的分组信息缓存至 Redis，因此双方发起事务，不需要确定连接到了哪个 TxManager，单一节点宕机可通过 Redis 轮询来确定 TxManager，达到事务管理的高可用性。LCN 分布式事务协调流程如图 4 所示。

在流程中，将所有服务单元的标识信息都存在统一单元内，并称为事务组。核心步骤分为 3 步：

（1）创建事务组：在服务发起方执行代码之前，创建事务协调者 TxManager，给服务分配 Group Id。

（2）添加事务组：在参与方执行结束代码之后，将本模块的信息发送给事务协调者。

（3）关闭事务组：在发起方执行了业务代码之后，将发起方状态发送给事务协调者。

（4）当关闭事务组之后，事务协调者根据事务组信息来通知参与方提交或回滚。

综上,LCN 框架能够解决在分布式系统下的事务一致性问题,并且事务操作和代码耦合度极低、通过集群化能够达到高可用,保证了大型公司在高并发场景下的事务完整性。

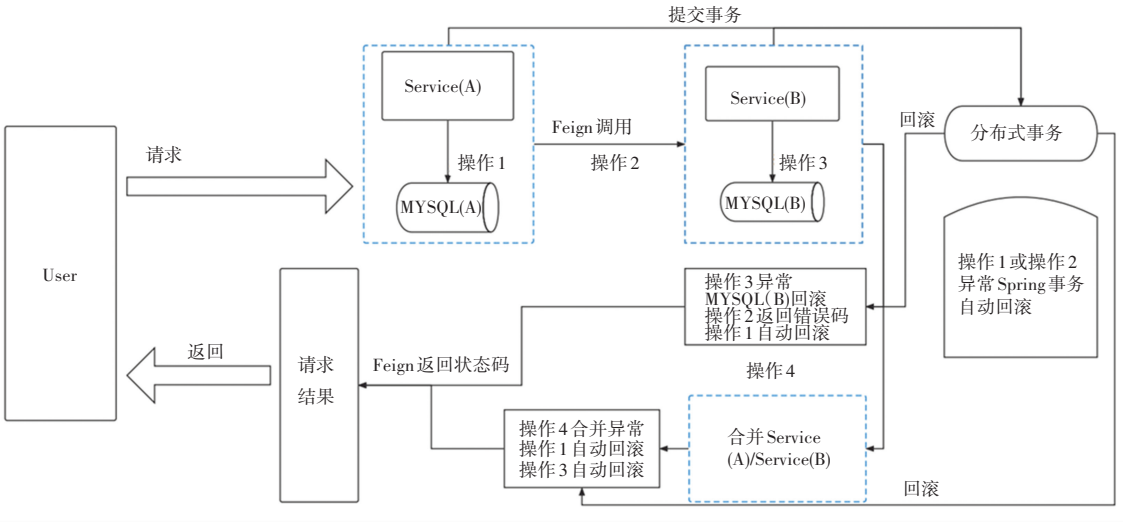

图 4 分布式事务流程图

Fig. 4 LCN distributed transaction

4 结束语

本文通过分析当下大宗商品 ERP 系统的开发现状,研究基于微服务架构对当前问题的解决方案,搭建基于微服务架构的大宗商品仓储管理系统,通过服务自动化管理、熔断限流、服务无状态化以及低耦合高兼容的事务管理实现了高并发、高可用、高安全性、高伸缩性以及事务一致性的分布式系统。业务部分针对木材大宗商品市场进行定制化的系统设计,并与公司业务深度结合,抛开固有的模板化系统设计,使系统具有更多的适用性与变通性。本系统现已投入使用并取得良好成果,为同类型企业的信息管理系统提供了优质的范本。

参考文献

[1] KHAN M G, HUDA N U l, ZAMAN U K U. Smart warehouse management system: Architecture, real-time implementation and prototype design[J]. Machines,2022,10(2).

[2] 李春阳,刘迪,崔蔚,等. 基于微服务架构的统一应用开发平台[J]. 计算机系统应用,2017,26(04):43-48.

[3] YANG Ning. Design and application of logistics warehouse management system in iron and steel enterprises under the background of computer big data [J]. International Journal of Intelligent Information and Management Science,2021,10(6).

[4] EBERT C, GALLARDO G, HERNANTES J, et al. DevOps[J]. IEEE Software, 2016, 33(3):94-100.

[5] 黄启启,项前,程茂上,等. 基于微服务的仓储管理与控制系统[J]. 东华大学学报(自然科学版),2020,46(01):83-90.

[6] 王艳清,陈红. 基于 SSM 框架的智能 web 系统研发设计[J]. 计算机工程与设计,2012,33(12):4751-4757.

[7] 万燕,朱翔. 微服务架构在校园智能安全接送系统中的应用[J]. 智能计算机与应用,2020,10(05):158-162,168.

[8] 周丹,雷晓玲,章民融. 基于微服务架构的校车安全管理系统设计与应用[J]. 计算机应用与软件,2018,35(08):165-169.

[9] 周鹏飞,张贺,魏磊,等. 基于微服务架构的整车制造仓储管理系统设计[J]. 物流技术,2021,40(05):107-111.

[10] DRAGONI N, GIALLORENZO S, LAFUENTE A L, et al. Microservices: Yesterday, today, and tomorrow[J]. Present and Ulterior Software Engineering, 2017(6): 195-216.

[11] NEWMAN S. Building microservices [M]. USA: O´Reilly Media, Inc., 2021.

[12] 洪华军,吴建波,冷文浩. 一种基于微服务架构的业务系统设计与实现[J]. 计算机与数字工程,2018,46(01):149-154.

[13] THÖNES J. Microservices[J]. IEEE Software, 2015,32(1):116.

[14] 方意,朱永强,宫学庆. 微服务架构下的分布式事务处理[J]. 计算机应用与软件,2019,36(01):152-158.

[15] Cosmina I. Spring microservices with spring cloud [M]// PivotalCertified Professional Spring Developer Exam. Berkeley, CA:Apress, 2017: 435-459.

[16] 李光明,延雄. 面向西门子 S7 系列 PLC 的智能管控系统[J]. 现代制造工程,2022(02):37-45.

[17] 霍福华,韩慧. 基于 SpringBoot 微服务架构下前后端分离的 MVVM 模型[J]. 电子技术与软件工程,2022(01):73-76.

[18] 赵昌建. 关于 SQL 数据库的性能优化问题的研究[J]. 科技风,2018(15):66.

[19] FERRAIOLO D, KUHN D R, CHANDRAMOULI R. Role-based access control[M]. USA:Artech house, 2003.

[20] 李孟珂,余祥宣. 基于角色的访问控制技术及应用[J]. 计算机应用研究,2000(10):44-47.

[21] 贾希旺,赵晓东,柳先辉. 基于微服务的智能制造知识图谱平台架构[J/OL]. 计算机集成制造系统:1-16[2022-03-14]. http://kns.cnki.net/kcms/detail/11.5946.TP.20220305.1536.002.html.

[22] 倪小璐,王旭英,边俊凯,等. 微服务软件架构设计模式及其应用[J]. 杭州师范大学学报(自然科学版),2021,20(04):442-448.